TECHNICAL EXPECTATIONS

Introduction

This project implements a classic Snake game using Python's Turtle graphics module. The game features a snake controlled by the player which grows longer by eating food items appearing in random positions. The game tracks the player's score and maintains a high score. It includes custom visuals such as a gradient background, snake eyes, and a flickering tongue for enhanced player experience.

Technical Design and Architecture

Modular Design

The original code was organized in a single script. To meet technical expectations, the project should be modularized into multiple files each responsible for discrete functionality. Key modules include:

- graphics.py: Responsible for drawing the gradient background and managing visual elements like the snake's eyes and tongue.
- snake.py: Manages the snake's head, body segments, movement logic including zigzag motion, and collision detection.
- food.py: Handles food objects and their random placement.
- ui.py: Displays the score and high score to the player.
- controls.py: Processes keyboard inputs and updates the snake's moving direction.
- main.py: The program entry point that sets up the game window, initializes objects, and controls the game loop.

This separation enhances maintainability, readability, and enables easier testing.

Application of Concepts

- Data Structures: The snake's body is represented by a list of segments which are dynamically increased when food is eaten.

- Algorithms: Collision detection algorithms handle checking boundaries and self-collision. Movement incorporates composite translation with zigzag offsets, adding complexity to directional changes.
- Design Patterns: The program uses event-driven design to respond to keyboard inputs.

Code Implementation Highlights

Visuals

- Gradient black background created with turtle fills simulates depth.
- Snake head customized with vibrant green color and outlined pen. It includes eyes and a red tongue that flickers while moving.
- Food items are represented by red circles or apple images if available, distributed initially with offsets to avoid overlap.

Controls

The snake moves in four directions—up, down, left, right—controlled via the 'W', 'S', 'A', and 'D' keys respectively, with direction restrictions to avoid reversing directly.

Game Logic

- The snake moves continuously with periodic zigzag offsets.
- Eating food causes the snake to grow, score to increment by 10, and game speed to gradually increase by decreasing delay time.
- The game resets if the snake hits the screen boundary or its own body.
- Score and high score display update accordingly on-screen.

Documentation and Comments

Each function and significant block of code is documented with comments describing purpose and logic. For example, the `update_eyes()` function positions the snake's eyes based on the current direction, improving player visual orientation.

Validation and Error Handling

- Robust handling of missing apple image assets with fallback to default circle shapes.
- Bounds checking to reset the game when the snake collides with screen edges.
- Segments are cleared appropriately on reset to avoid residue artifacts.
- Delay time is clamped to a minimum threshold to prevent the game from becoming too fast.

Testing

While the current script runs as a standalone, converting to modular components allows unit testing. For example, movement, collision, and scoring can be tested separately using Python's `unittest` framework by mocking turtle objects.

Version Control

The project is recommended to be maintained under Git version control for incremental development tracking and collaboration readiness. Commits should document feature additions and bug fixes clearly.
